HUFFBIT COMPRESS – ALGORITHM TO COMPRESS DNA SEQUENCES USING EXTENDED BINARY TREES.

P.RAJA RAJESWARI [1] Dr. ALLAM APPARAO [2] Dr. R.KIRAN KUMAR [3]

(1) Research Scholar , Acharya Nagarjuna University,Guntur.
(2) Vice-chancellor ,Jawaharlal Nehru Technological University, Kakinada.
(3) Computer science and Engineering, Krishna University.
Email: rajilikhitha@gmail.com, raji_likhitha@yahoo.com

ABSTRACT

Compressing DNA sequences is an important task as every day thousands of gigabytes of sequences of nucleotides and amino acids gets stored in Genbank. Storing large Genomes in a personal computer in the compressed form is an efficient means of using DNA sequences for biological functions.

We present a compression algorithm, "HuffBit Compress" for DNA sequences based on a novel algorithm of assigning binary bit codes(0 and 1) for each base(A,C,G,T) to compress both repetitive and non repetitive DNA sequence. Our proposed algorithm achieves the compression ratio using the concept of Extended Binary Tree. In this algorithm Extended binary tree concept is applied to derieve a special class of variable-length codes that satisfy prefix property. This class of codes is called Huffman Codes. Huffman codes are based on relative frequency with which DNA symbols (A,C,G,T) appear in the sequence.

The specific bit sequence assigned to an individual base is determined by tracing out the path from the root of the tree to the leaf that represents that Base. By convention, bit '0' represents following the left child when tracing out the path and bit '1' represents following the right child when tracing out the path. The extended binary tree is constructed such that the paths from the root to the most frequently used bases are short while the paths to less frequently used bases are long. This results in short codes for frequently used bases and long codes for less frequently used bases.

If the sequences are encoded using the old trivial method of assigning 2 bits per base.(A=00,C=01,G=10,T=11) The encoded text is of higher length.(1000 bases needs 2000 bits of space). This proposed algorithm "HuffBit Compress" compresses DNA sequences more efficiently.(1000 bases may need only 1006 bits of space in Best case.). Analysis of Best case and worst case is defined for the first time in this Algorithm. Compression Ratio of 1.6 bits/base can be achieved using this proposed algorithm. This algorithm makes the near choice (minimum value) at every step that appears to lead to an overall optimal base encoding.

KEYWORDS: *Huffman Codes, Compression Ratio, Internal Node, External Node, Extended Binary Tree, Encode, Decode.*

1: INTRODUCTION:

In modern molecular biology, the **genome** is the entirety of an organism's hereditary information. It is encoded either in DNA or, for many types of virus, in RNA. The genome includes both the genes and the non-coding sequences of the DNA.(Ridley, M. (2006) *Genome)*. Increasing genome sequence data of organisms lead DNA database size two or three times bigger annually. Thus it becomes very hard to download and maintain data in a local system.

1.1 EXISTING ALGORITHM

For a four-letter alphabet in DNA(A,C,G,T),an average description length of 2 bits per base can be assigned . For the string A,C,G,T whose sequence length is 1000, assigning A=00,C=01,G=10,T=11, requires 2000 bits of space. The storage of encoded sequence is almost double its original sequence length[1].

2. PROPOSED ALGORITHM - HUFFBIT COMPRESS

The proposed algorithm is a 2 way process :

(a) First construction of extended binary tree is done
(b) Second, The derived Huffman codes from Extended Binary Tree are used to calculate the number of bits in the encoded sequence length. Compression ratio is compressed size by uncompressed Size.

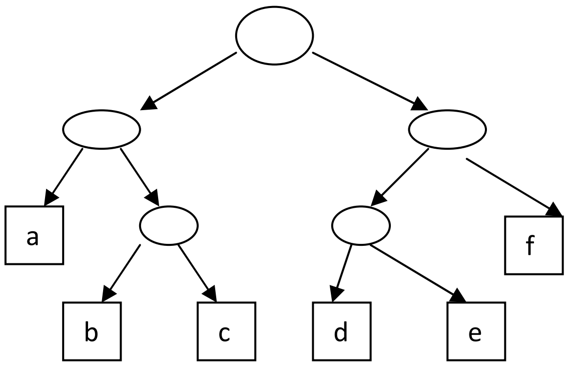
2.1 : EXTENDED BINARY TREE

An extended binary tree is a transformation of any binary tree in to a complete binary tree.

This transformation consists of replacing every null subtree of the original tree with "special nodes".

The nodes from the original tree are called internal nodes. [2]The special nodes are called external nodes The following tree is extended binary tree. Empty circles represent internal nodes and boxes represent External nodes. Every internal node consists of exactly 2 children and every external node is a leaf.[3]

FIGURE 1: Extended Binary Tree



2.1.1:EXTENDED BINARY TREE FOR DNA SEQUENCE {A,C,G,T}

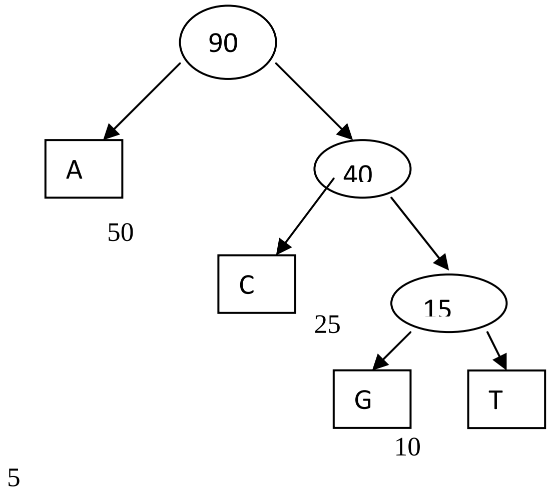
Consider $\partial(x)$ be the length of a shortest path from node x to an external node in its subtree. Suppose if x is an external node, x is assumed as "0".
If x is an internal node, ∂ value is :
Internal node ∂ = Min{$\partial[\tau]$,$\partial[R]$} + 1.
Where τ = left child of x.
and R = right child of x.

Example :

Given Sequence = a,c,g,t.
length = 4.

Number of internal nodes = 3
Number of external nodes = 4.

FIGURE 2: Extended Binary tree for DNA bases A,C,G,T.

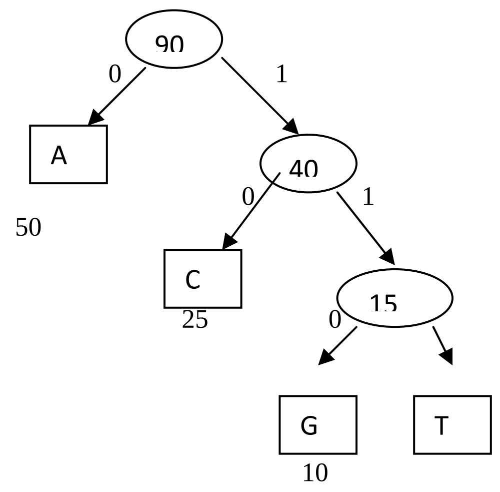


2.2: HUFFMAN CODING

The basic idea of Huffman coding is simple.[4] We take some data we want to compress, say a list of 8 bit characters. We then create a value table where we order the characters by frequency. If we don't know beforehand how our list of characters will look, we can order the characters by probability of occurring in the string. We then assign code words to the value table, where we assign the short code words to the most probable values. A code word is simply an n-bit integer designed in such a way there are no ambiguities or clashes with shorter code words[5]

Using the figure of a,c,g,t extended binary tree, By convention, bit '0' represents following the left child when tracing out the path and bit '1' represents following the right child when tracing out the path.

FIGURE 3: Bases with their Huffman Codes.



1

5

The huffman codes for A,C,G,T travelling from root to their external node.
A=0
C=10
G=110
T=111

Let "s" be a sequence of bases.
F(x) be the frequency of bases , then x ξ {a,c,g,t}

Weighted Extended Path = $\sum_{i=1}^{n} \lambda(I) * F(I)$

Where
λ(I) = length of path (number of edges on path) from root to external node labeled i.
F(I) = Frequency of occurances of bases (a,c,g,t).

Calculation of Bits in Encoded sequence =

$$\boxed{1*f(a)+2*f(c)+3*f(g)+3*f(t).}$$

Where λ(I) for base a= 1 (since A=0)
λ(I) for base c= 2 (since C=10)
λ (I) for base g= 3 (since G=110)
λ (I) for base t= 3 (since T=111)

2.3 : ENCODE ALGORITHM

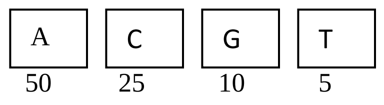
- ❖ Determine the different bases in the given DNA sequence(a,c,g,t) and calculate their Frequencies.
- ❖ Construct a Extended Binary Tree with minimum Weighted path(Huffman Tree).
- ❖ The External nodes of this tree are labeled by the bases in the sequence.
- ❖ The weight of each External node is the frequency of the base and that is its label(A,C,G,T).
- ❖ Traverse the root-to-external node paths and obtain the codes.
- ❖ Replace the symbols in the sequence by their codes(Huffmans Codes).

2.4: CONSTRUCTION OF HUFFMAN TREE TO MINIMIZE THE LENGTH OF CODED SEQUENCE:

- ➢ Use codes from a binary tree whose external nodes correspond to the bases in the sequence
- ➢ being encoded.
- ➢ Select the external nodes whose Weighted External Path is minimum.
- ➢ The binary tree with minimum Weighted External Path for a given set of frequencies is called a Huffman tree.

FIGURE 4: Construction of Huffman Tree

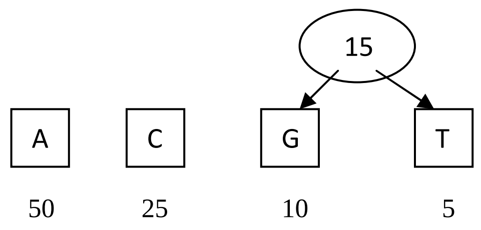
(a)



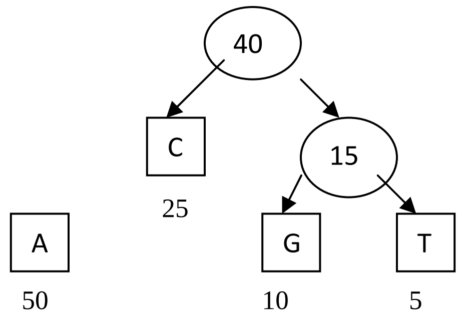
Frequency

Combining the lowest frequency values.

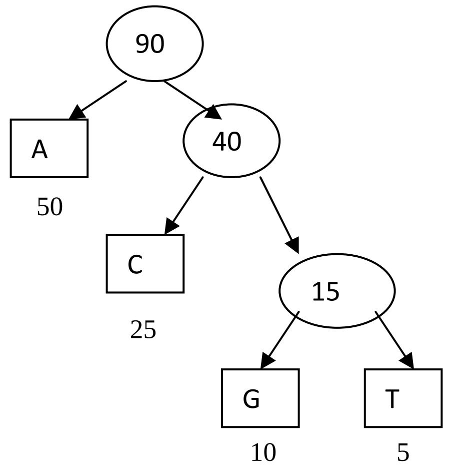
(b)



(c)



(d)



2.4: Decode Algorithm.

Decoding is the reverse of coding.[6] If we have a bit string, say 00011111010, we read bits until there's a match in the table. Our example string decodes to AAATGC. Note that the code word table is designed so there are no conflicts. If the table read A=0,C=10,G=110,T=111.and we encounter the bit 0 in a table, there's no way we can ever get a C as the A will match all the time.
The standard method of decoding a Huffman coded string is by walking a binary tree, created from the code word table. When we encounter a 0 bit, we move left in the tree, and right when we see a 1. This is the simplest method used in our decoder.

- ❖ Decoding of binary bits leads to the original base sequence.
- ❖ The codes (0,10,110,111) are unique and no code is a prefix of another.
- ❖ Consequently when the code is examined from left to right a match with exactly one code
 is obtained, hence decoding is possible using Huffman's code.
- ❖ No root-to-external node path is a prefix of another such path.
- ❖ No path code is a prefix of another path code.

3: EXAMPLES AND CALCULATIONS

3.1: BEST CASE:

Given sequence :
aaaaaaaaaaaaaaaaaaaaaaaaaccaaaaaaaaaaaaaaaagaa aaaaaaaaaaaaat…………………
Sequence length = 1000.

Frequency of base "A " in the given sequence = 996
Frequency of base "C " in the given sequence = 2
Frequency of base "G " in the given sequence = 1
Frequency of base "T " in the given sequence = 1

Huffman codes for bases A=0,C=10,G=110, and T=111.

Number of Bits in Encoded sequence
= 1*f(a)+2*f(c)+3*f(g)+3*f(t)
= 1*996 + 2* 2 + 3*1 + 3*1
= 996+4+3+3
= 1006 bits.

Compression Ratio = number of bits encoded / number of bases.
= 1006/1000
= 1.006 bits/base.

3.2: AVERAGE CASE:

Given sequence :

Sequence length = 90.

Frequency of base "A " in the given sequence = 50
Frequency of base "C " in the given sequence = 25
Frequency of base "G " in the given sequence = 10
Frequency of base "T " in the given sequence = 5

Huffman codes for bases A=0,C=10,G=110, and T=111.

Number of Bits in Encoded sequence
= 1*f(a)+2*f(c)+3*f(g)+3*f(t).
= 1*50 + 2* 25 + 3*10 + 3*5
= 50+50+30+15
= 145 bits.

3.3: WORST CASE:

Let us consider the sequence:
GAAT TTGC AAAA AAAA GCTA ATGC CTAG GGTT TTTG CCCC CCCC AAAA TCAG TTGC ATAG GACG .

Sequence Length = 64.

Frequency of base “A “ in the given sequence = 21
Frequency of base “C “ in the given sequence = 15
Frequency of base “G “ in the given sequence = 13
Frequency of base “T “ in the given sequence = 15

Huffman codes for bases A=0,C=10,G=110, and T=111.

Number of Bits in Encoded sequence
= 1*f(a)+2*f(c)+3*f(g)+3*f(t).
= 1*21 + 2* 15 + 3*13 + 3*15
= 21+30+39+45
= 135 bits.

Compression Ratio = number of bits encoded / number of bases.
= 135/ 64
= 2.109 bits/base.

Compression Ratio = number of bits encoded / number of bases.
= 145/ 90
= 1.611 bits/base

4: CONCLUSION

A simple DNA compression algorithm using the concept of Extended Binary Tree is proposed to compress DNA sequences which are repetitive as well as non repetitive in nature. The extended binary trees are used to derive the variable length codes that satisfy the prefix property. Since Huffmans code satisfy Prefix property, its an efficient way to encode and decode DNA sequences. HuffBit algorithm uses the technique of Greedy method. DNA sequence analysis i.e. single and multiple alignments are areas of active research in bioinformatics. If the sequence is compressed using HuffBit Compress algorithm, it will be easier to compress large bytes of DNA sequences with good compression ratio. Average compression ratio that can be achieved using this algorithm is 1.6 bits/base. The Time complexity is given by O(n log n).

5: LIMITATIONS

Testing the real biological sequences on this alogrithm, performance with the tools, or the transfer of knowledge to similar tasks could not be performed.

6: FUTURE WORK

Developing a java based tool for HuffBit compress algorithm and to test for compression ratios of real DNA sequences of Large Genomes.

REFERENCES

[1] A New Challenge for Compression Algorithms: Genetic Sequences by Grumbach, Stephane; Tahi, Fariza

[2] Left and right length of paths in binary trees or on a question of knuth. By Alois Panholzer

[3] Meta-Fibonacci sequences, Binary Tees and Extremal Compact Codes by B.Jackson
And Frank Ruskey.

[4] On the competitive optimality of Huffman codes by Thomas. M. Cover.

[5]Two algorithms for constructing efficient huffman-code based reversible variable length codes
Chia-Wei Lin; Ja-Ling Wu; Yuh-Jue Chuang

[6]Guaranteed Synchronization of Huffman Codes with Known Position of Decoder
Marek Tomasz Biskup, Wojciech Plandowski,

P.Raja Rajeswari received her post graduate degree in Computer Applications in 1999 and M.Tech[IT] in 2003. She is working as Assistant Professor in DMSSVH college of Engineering ,Machilipatnam since 2000 to till date.She is pursuing her Ph.D from Acharya Nagarjuna University in Computer Science under the guidance of Dr. Allam Appa Rao. Her research interests includes Bioinformatics, compression techniques, design and analysis of Algorithms,development of software tools

Dr. Allam Appa Rao has received PhD in Computer Engineering from Andhra University, Visakhapatnam, Andhra Pradesh, India.He has worked as the Professor in Bioinformatics & Computational Biology, Department of Computer Science and Systems Engineering &Principal, Andhra University College of Engineering (AUTONOMOUS). Currently he is Vice Chancellor to Jawaharlal NehruTechnological University, Kakinada. His research interest includes Bioinformatics, Software Engineering and Network Security. He is a member of professional societies like IEEE, ACM and a life member of CSI and ISTE. www.allamapparao.net.

Dr. Kiran Kumar Reddi has received PhD in Computer Science and Engineering from Acharya Nagarjuna University, Guntur, Andhra Pradesh, India.He is working as Assistant Professor in the department of Computer Science, Krishna University, Machilipatnam, Andhra Pradesh, India. His research interest includes Bioinformatics, Software Engineering and Network Security. He is a member of professional societies like CSI and ISTE.